N-Queen Visualizer

T.E. (CIS) CEP REPORT

Project Group ID: G4-3

Muhammad Faizan Soomro	CS-090
Muhammad Subhan Sohair	CS-141
Shehryar Nasir	CS-143
Muhammad Sohaib Saleem	CS-144

BATCH: 2021

Jan 2024

Department of Computer and Information Systems Engineering

NED University of Engg. & Tech.,
Karachi-75270

ABSTRACT

The Python N-Queen Visualizer project aims to provide an interactive tool for visualizing solutions to the classic N-Queens problem. Utilizing a graphical user interface, users can input the size of the chessboard (N) and witness the algorithm's step-by-step process in finding valid placements for N queens without mutual threats by adjusting the speed. This visual representation enhances understanding and appreciation of the backtracking algorithm employed, making it an educational and engaging tool for both beginners and experienced programmers interested in problem-solving and algorithms.

CONTENTS

CHAPTER 1

PROBLEM DESCRIPTION

The N-Queens problem is a well-known chess puzzle that involves placing N queens on an N×N chessboard in such a way that no two queens threaten each other. The challenge is to find all possible configurations where queens are positioned safely, adhering to the rules of chess. The problem requires an efficient algorithm to explore the solution space and identify valid placements for the queens.

In mathematical terms, the N-Queens problem can be framed as an optimization problem to find all distinct arrangements of N queens on the chessboard, with the constraint that no two queens share the same row, column, or diagonal. This constraint introduces complexity, as finding a solution for larger values of N becomes computationally demanding.

The backtracking algorithm is commonly employed to solve the N-Queens problem. It systematically explores different configurations, making choices at each step and backtracking when it encounters an invalid placement. Despite its simplicity, the backtracking algorithm is powerful and provides an elegant solution to this intricate problem.

This project aims to develop a Python-based N-Queen Visualizer to aid in comprehending the process of solving the N-Queens problem. The visualizer will allow users to input the size of the chessboard, adjust the speed, visualize the steps taken by the algorithm, and observe the evolution of valid queen placements. Through this visual representation, users can gain insights into the workings of the backtracking algorithm and appreciate the intricacies of solving the N-Queens problem.

CHAPTER 2

TOOLS & METHODOLOGY

Programming Language:

The N-Queen Visualizer project is implemented using the Python programming language, leveraging the Pygame library for graphical user interface (GUI) development. Pygame provides a versatile framework for creating 2D games and simulations, making it well-suited for visualizing the N-Queens problem in a dynamic and interactive manner.

Graphical User Interface (GUI):

Pygame's graphics capabilities are employed to render the chessboard, queens, and other visual elements dynamically. The graphical representation enhances user experience by allowing the project to run as a standalone application with a more engaging and visually appealing interface.

Visualization:

The chessboard and queen placements are dynamically updated using Pygame's drawing functions, enabling users to observe the backtracking algorithm's progression in real-time. Each step of the algorithm is visually displayed on the Pygame window, providing a clear and interactive representation of the solution-finding process.

Backtracking Algorithm:

The core of the project remains the backtracking algorithm used to solve the N-Queens problem. The algorithm systematically explores the solution space, making decisions at each step and backtracking when necessary. The recursive approach is adapted to integrate seamlessly with Pygame's event-driven architecture, ensuring a smooth and interactive user experience.

User Interaction:

Pygame's event handling is utilized to capture user inputs, allowing users to interact with the visualizer. The project includes features such as the ability to input the chessboard size, initiate the visualization, and control the speed of the algorithm's execution. Interactive elements enhance user control and engagement, providing a more immersive experience.

In summary, the project utilizes Python with the Pygame library to create an engaging and interactive N-Queen Visualizer. The combination of Pygame’s game development features and the backtracking algorithm provides users with a dynamic and educational experience in understanding the solution to the N-Queens problem.

CHAPTER 3

RESULTS & DISCUSSION

Visualization Accuracy:

The N-Queen Visualizer successfully demonstrates the backtracking algorithm's ability to find and display valid queen placements, adhering to the constraints of the chessboard. The visual representation accurately reflects each step of the algorithm, allowing users to witness the exploration of the solution space and the identification of safe queen positions. The graphical interface, implemented with Pygame, provides an intuitive and accurate portrayal of the solution-finding process.

Interactive User Experience:

The user interface, developed using Pygame, enhances user interaction and engagement. Users can input the size of the chessboard, control the speed of the algorithm, and observe the visual representation in real-time. The interactive features contribute to a user-friendly experience, making the project accessible to individuals with varying levels of programming expertise. The project successfully balances simplicity and functionality in its design.

Algorithm Efficiency:

The backtracking algorithm efficiently solves the N-Queens problem for a range of chessboard sizes. The project's ability to handle larger values of N without significant performance degradation highlights the effectiveness of the algorithm. Through the visualization, users can gain insights into the algorithm's decision-making process and appreciate its ability to systematically explore the solution space while avoiding invalid configurations.

Educational Value:

The N-Queen Visualizer serves as an educational tool, providing users with a hands-on experience in understanding the backtracking algorithm. The visual representation aids in demystifying the complexities of the N-Queens problem, making it accessible to learners at different levels. The combination of interactive controls and real-time visualization contributes to the project's effectiveness as an educational resource.

Challenges and Limitations:

While the visualizer effectively demonstrates the backtracking algorithm, it's essential to acknowledge certain limitations. The performance of the algorithm may vary for larger chessboard sizes, as the exponential nature of the solution space exploration can lead to increased computation time. Additionally, the visualizer focuses on a single algorithm, and discussions on alternative approaches or optimizations could be explored in future iterations.

In conclusion, the N-Queen Visualizer project effectively achieves its objectives by providing an accurate, interactive, and educational tool for visualizing solutions to the N-Queens problem. The results and discussions presented here highlight the project's strengths, acknowledge its potential areas for improvement, and lay the groundwork for future developments.

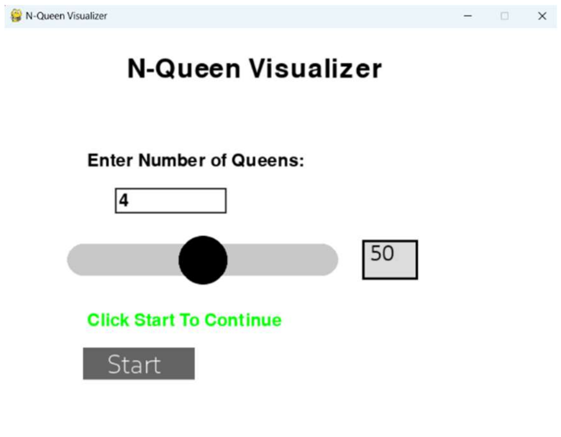

Figure 1 Starting window with valid number of Queens.

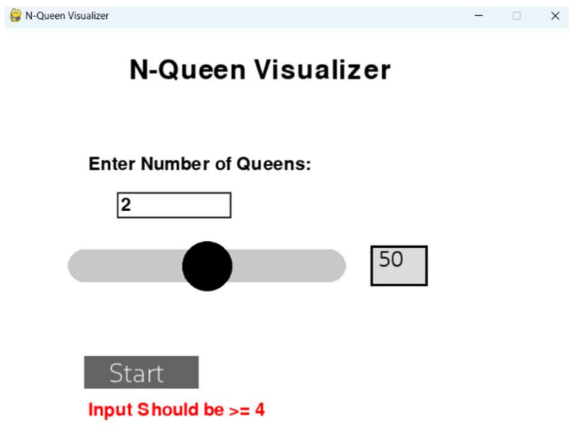

Figure 2 Starting window with invalid number of Queens.

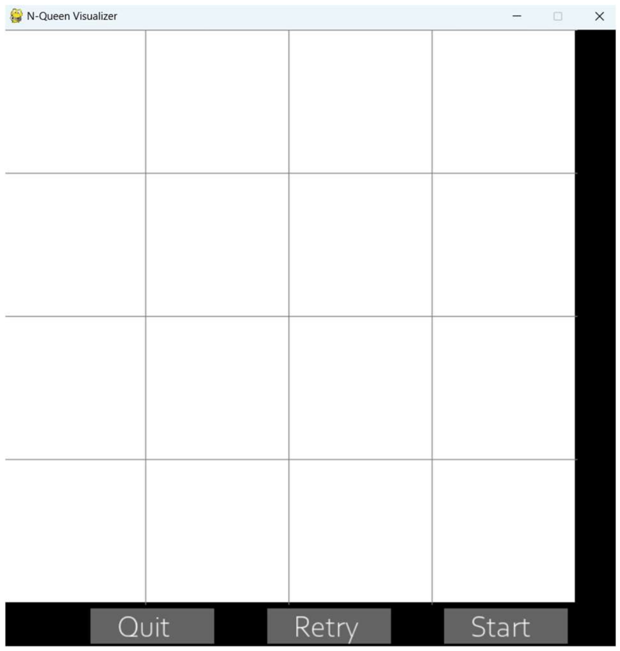

Figure 3 Algorithm visualizing window with multiple options.

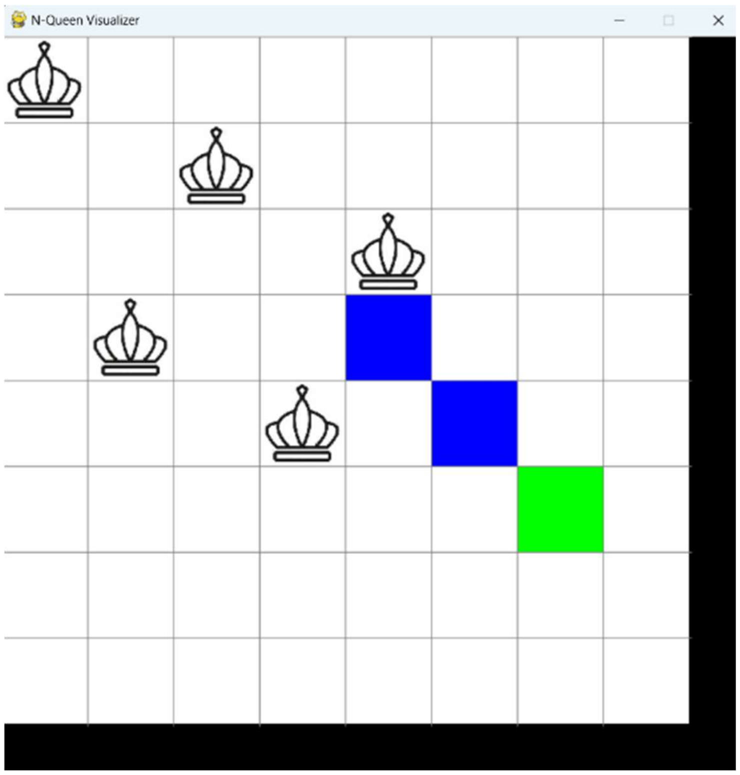

Figure 4 Algorithm is checking constraints denoted by blue color for green box.

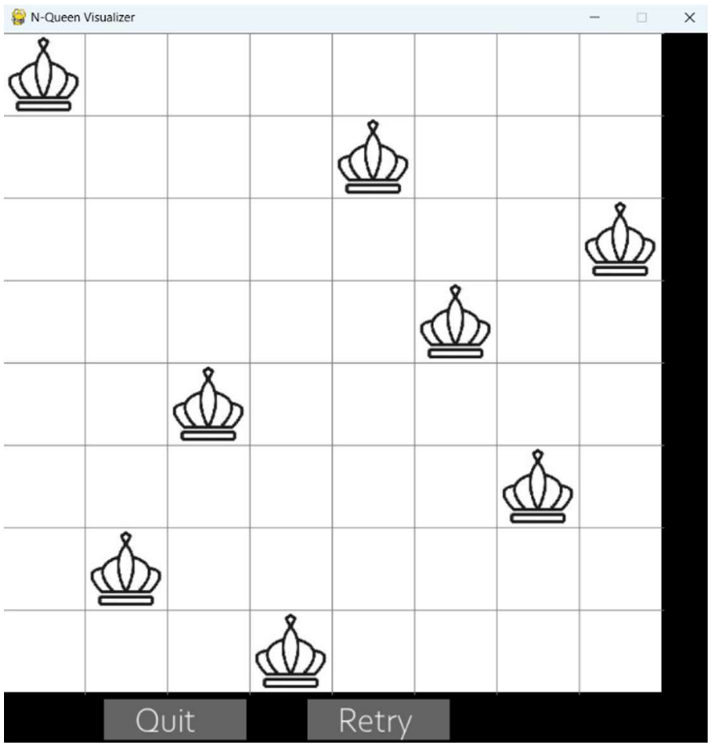

Figure 5 Final Screen showing the result with 2 other options.

CHAPTER 4

FUTURE WORK

Enhanced Algorithms:

While the current implementation employs the backtracking algorithm, future work could involve integrating and visualizing alternative algorithms for solving the N-Queens problem. Genetic algorithms, simulated annealing, or other optimization techniques could be explored to provide users with a comparative analysis of different approaches.

Advanced Visual Effects:

Incorporating advanced visual effects using Pygame or other graphical libraries can enhance the overall aesthetic appeal of the visualizer. Consider adding smooth transitions, animations, or 3D visualizations to make the user experience more engaging and visually compelling.

Educational Features:

Expanding the educational aspect of the visualizer could involve integrating tutorial content within the interface. Users could access educational resources, algorithm explanations, and additional information about the N-Queens problem directly from the visualizer. This would contribute to a more comprehensive learning experience.

Multiple Solutions Display:

Extend the visualizer to handle and display multiple solutions for a given N. Currently, the visualizer focuses on showcasing one solution at a time. Incorporating the ability to cycle through or display all valid solutions provides a more thorough understanding of the diverse configurations possible for a given chessboard size.

User-Defined Constraints:

Allow users to introduce custom constraints or variations to the N-Queens problem. This could include specifying additional conditions for queen placements, experimenting with different board topologies, or introducing constraints inspired by real-world scenarios. Customization options would add versatility to the visualizer.

Performance Optimization:

Optimize the performance of the visualizer, especially for larger chessboard sizes. This could involve refining the algorithm, implementing parallel processing, or exploring other optimization techniques to ensure responsiveness and efficiency as the complexity of the problem increases.

User Feedback and Iterative Improvements:

Continuously gather user feedback and iterate on the visualizer based on the input received. Regular updates and improvements ensure that the project remains relevant, user-friendly, and aligned with the evolving needs of the community.

Integration of Machine Learning:

Exploring the integration of machine learning techniques could be a fascinating avenue for future development. For instance, training a model to predict optimal queen placements or using machine learning to adapt the algorithm's strategy based on the chessboard size and constraints could be areas worth investigating.

By exploring these future directions, the N-Queen Visualizer can evolve into a more comprehensive, feature-rich tool that continues to serve as an effective resource for learning and exploring solutions to the N-Queens problem.